A Variant of the Sum Coloring Problem on Trees

Chin-Fu Lin, Sheng-Lung Peng*, Min-Feng Wu
Department of Computer Science and Information Engineering
National Dong Hwa University, Hualien 97401, Taiwan
*corresponding author: slpeng@mail.ndhu.edu.tw

Abstract

Let $G = (V, E)$ be a simple and connected graph. The graph coloring problem on G is to color the vertices of G such that the colors of any two adjacent vertices are different and the number of used colors is as small as possible. By assigning a positive integer to a color, the sum coloring problem asks the minimum sum of the coloring numbers assigned for all vertices. In this paper, we introduce a new coloring problem called the distinguishable sum coloring problem. In this problem, one additional condition is required, i.e., *all the vertices in the closed neighborhood of any vertex must be colored in different colors. We propose a dynamic programming algorithm for solving the distinguishable sum coloring problem on trees. The time complexity of this algorithm is $O(n \times \Delta(T)^2 \times \Delta(T)!)$, where T is a tree, $n = |V|$, and $\Delta(T)$ is the maximum degree of T. Also, we obtain a recurrence relation for this problem on full k-ary trees. Note that if $\Delta(T)$ is constant, then our algorithm runs in $O(n)$ time.*

1 Introduction

Let $G = (V, E)$ be a simple and connected graph. Let $N(v) = \{u \mid (u, v) \in E\}$ be the open neighborhood of v and $N[v] = N(v) \cup \{v\}$ be the closed neighborhood of v. The degree of a vertex v is denoted by $deg(v)$ which is equal to $|N(v)|$. The maximum degree of a graph G is denoted by $\Delta(G)$.

A coloring on G is a function f on V mapping to $\{1, 2, \ldots, k\}$ such that for any two adjacent vertices u and v, *i.e.*, $(u, v) \in E$, $f(u) \neq f(v)$. A coloring using at most k colors is called a k-coloring. The **Graph Coloring Problem** (GCP for shot) on G is to find a coloring function f such that k is minimum. The k is called the `chromatic number` of G and is denoted as $\chi(G)$.

GCP is a classic NP-Hard problem [6] and has many applications such as scheduling [11], timetabling [14], frequency assignment [16], printed circuit testing [5], register allocation [17], communication networks [18] and bag rationalization [7]. On the other hand, current algorithms can only solve small random graphs, with up to 80 vertices [3]. However, today's applications usually need more than hundreds or thousands of vertices. Therefore, some heuristic and metaheuristic algorithms have been proposed [12, 15].

The **Minimum Sum Coloring Problem** (MSCP for short) on G is to find a coloring function f such that $\sum_{v \in V} f(v)$ is minimum. This minimum sum of colors is called the `chromatic sum` of G and is denoted by $\sum(G)$. The number of colors needed in $\sum(G)$ is called the `strength` of G. It seems that $\chi(G)$ colors are sufficient for the MSCP. However, it has been proved that the strength of a graph G sometimes needs more than $\chi(G)$ colors. Figure 1 uses $\chi(G)$ colors to color a tree with $\sum_{v \in V} f(v) = 12$. However, Figure 2 uses extra 1 color to color the same tree with $\sum_{v \in V} f(v) = 11$.

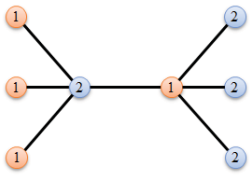

Figure 1: Coloring a tree with sum 12 using 2 colors.

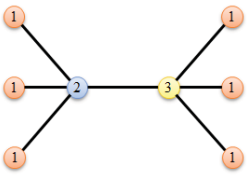

Figure 2: Coloring the same tree with sum 11 using 1 extra color.

The research of the MSCP is started from Kubicka in 1989 [8]. Kubicka proved that the MSCP was NP-complete and gave a polynomial time algorithm on trees. Kroon [10] proved the NP-hardness of MSCP on interval graphs and gave a linear time algorithm for trees. Jansen showed

that the MSCP for cographs can be solved in $O(|V| + |E|)$ time and the partial k-trees can be solved in $O(|V| \log^{k+1} |V|)$ time. He also proved that this problem on bipartite graphs and permutation graphs was NP-complete. In addition, he also proved that the MSCP problem can be solved for cobipartite graphs in polynomial time. Besides, several heuristic algorithms have been proposed [1, 2, 4, 9, 13, 19]. A distinguishable coloring on G is a function f on V mapping to $\{1, 2, \ldots, k\}$ such that $|\{f(u) \mid u \in N[v]\}| = |N[v]|$ for each $v \in V$. It means that any vertex and all its neighbors must be colored different from each other. The **Distinguishable Sum Coloring Problem** (DSCP for short) on G is to find a distinguishable coloring function f such that $\sum_{v \in V} f(v)$ is minimum.

2 An algorithm for trees

In this section, we propose an algorithm to solve the DSCP on trees. We define the T_d^k to be a rooted full k-ary tree with depth d. If $k = 2$, then it is a full binary tree. Let $sum(T) = \sum_{v \in V} f(v)$.

It is easy to check that $sum(T_0^k) = 1$ and $sum(T_1^k) = \frac{(k+1)\times(k+2)}{2}$. For the solution of T_1^k, it can be considered as a star graph with $k+1$ nodes. So, we need the colors from 1 to $k+1$. It can also be considered as the root colored 1 with the second best of $T_0^k + 1$, the third best of $T_0^k + 2, \ldots$, and so on. Note that all vertices can be exchanged with each other.

For T_2^k tree, we can decompose it into k T_1^k trees. By a similar argument, we can obtain that the $sum(T_2^k)$ is $k \times sum(T_1^k) + (k + 2)$. This is shown in Figure3. Note that every root vertex of T_1^k has a different color among these k T_1^k trees.

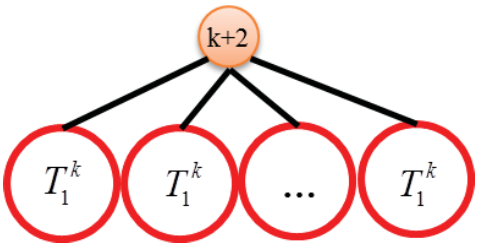

Figure 3: A solution for T_2^k.

By a similar argument, we can solve the problem on T_3^k. In this case, it is a little bit complicated. At first, we obtain a relation between the root color and the total sum as shown in Table 1.

Table 1: The relationship between the root value and the total sum of the tree.

root	$\cdots$	-2	-1	x	+1	$\cdots$	$+k$
sum	$\cdots$	$+(2k-2)$	$+(k-1)$	0	+1	$\cdots$	$+k$

Therefore, for the solution of T_3^k, the root can be colored with 1 and the remaining parts are T_2^K, $T_2^K + 1, \ldots, T_2^K + (k-1)$. Note that $T_2^K + (k-1)$ also can be changed into $T_2^K - 1$, but sometime $T_2^K - 1$ is not a good choice.

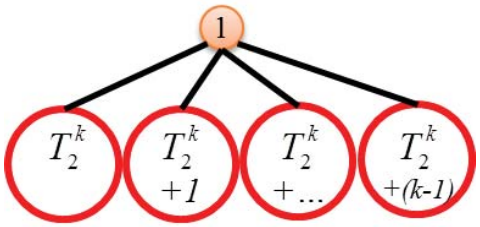

Figure 4: A solution of thee T_3^k.

Similarly, the solution of T_4^k is that the root can be colored 1 and the remaining parts are $T_3^K + 1$, $T_3^K + 2, \ldots, T_3^K + k$. Note that the root and the level-1 vertices can be exchanged with each other.

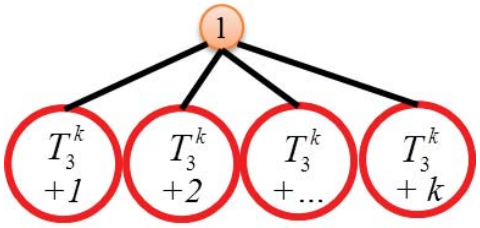

Figure 5: A solution of tree T_4^k.

Any T_d^k tree can be decomposed into a set of T_{d-1}^k trees by using the method mentioned above. Hence, the solution of a full k-ary tree, $sum(T_d^k)$, has the following possible cases.

$$\begin{cases} 1 & d = 0 \\ k \times sum(T_{d-1}^k) + \frac{k\times(k-1)}{2} + 1 & d > 0 \text{ and } d \mod 3 = 0 \\ k \times sum(T_{d-1}^k) + \frac{k\times(k+1)}{2} + 1 & d > 0 \text{ and } d \mod 3 = 1 \\ k \times sum(T_{d-1}^k) + (k+2) & d > 0 \text{ and } d \mod 3 = 2 \end{cases}$$

In the following, we propose a dynamic programming algorithm to solve the DSCP on general trees. By considering the input tree as a rooted tree, our algorithm works from bottom (leaves) to top (the root) to compute an optimal solution.

We first consider a leaf node v. Assume its parent is u. The initial value of v is easy since the maximum value of v is $|N[u]|$. For any internal vertex, we compute the minimum sum of its child vertices. Therefore, we must compute all the possible combinations from each vertex. For each internal vertex v, we must compute $\deg(v)^2$ times. Not only this vertex but also its parent vertex will influence the values of its child vertices. For example, in Figure 6, the table for the internal vertex v stores all the possible combinations for v and its parent. For example, the pair (3,4) means that v is colored with 3 and its parent is colored with 4. In this case, the child vertices cannot use the values 3 and 4. The minimum of this pair is $sum(\{1, 2, 5\}) = 8$.

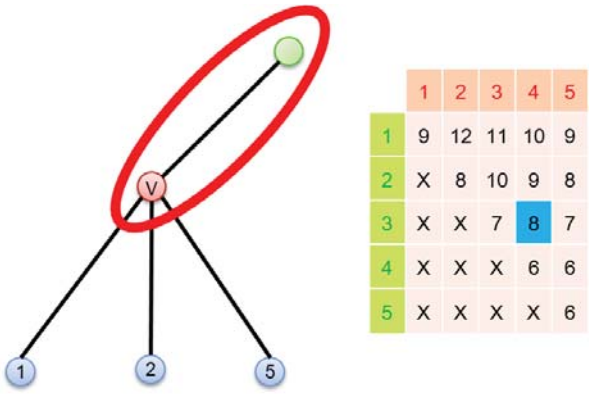

Figure 6: An example of the table for an internal vertex v.

If the child vertices are not leaves, then we must look for every child vertex's table. Consider Figure 7 as another example. The value of pair (4,3) is the minimum sum of the combination of v's child vertices with their tables. Due to the value of v is 4, we must look for row 4. If A's value is 1, then we look for the pair (1,4) of the table of vertex A. Note that if A's value is more than 5, then we look for the pair (4,4).

The following is the detail of our algorithm (Algorithm 1). We assume that we have enough information about the tree, i.e., each vertex knows all of its child vertices and its parent vertex. Also all the leaf vertices are considered that they have been computed in advance.

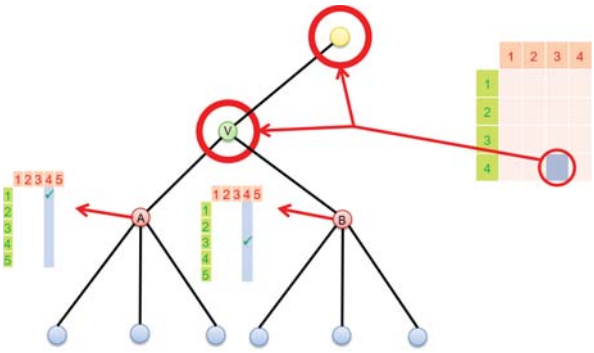

Figure 7: Another example for computing the table for an internal vertex.

Algorithm 2
Data: A tree $T = (V, E)$
Result: An optimal solution for DSCP on tree T

```
Let the root of T be r;
Create an empty stack S, and push r into S;
while S is not empty do
    Pop one node v from S;
    if the table of each v's child vertex is computed then
        Computed the the table of v by trying all possibilities;
    else
        Push v back to S;
        Push each child of v into S;
return the best value stored in the table of r;
```

Clearly, n vertices need $O(n)$. Each internal vertex needs to compute $\Delta(G)^2$ times and each time has $\Delta(G)!$ combinations. Thus, the time complexity of this algorithm is $O(n \times \Delta(G)^2 \times \Delta(G)!)$. If $\Delta(G)$ is constant, then our algorithm runs in $O(n)$ time.

3 Conclusion

In this paper, we propose a new coloring problem, namely, the distinguishable sum coloring problem. We propose a recurrence relation for solving the distinguishable sum coloring problem on full k-ary trees. Finally, we propose a dynamic programming algorithm on trees for the distinguishable sum coloring problem in time of $O(n \times \Delta(G)^2 \times \Delta(G)!)$. If $\Delta(G)$ is constant, then

it runs in $O(n)$ time. Is it possible to design a fully polynomial time algorithm for the distinguishable sum coloring problem on trees? It will be a future work.

Acknowledgements

This work was partially supported by the Ministry of Science and Technology of Taiwan, under contract MOST 103-2221-E-259 -030.

References

[1] A. Bar-Noy, M. Bellareb, M.M. Halldorsson, H. Shachnai and T. Tamir, On Chromatic Sums and Distributed Resource Allocation, *Information and Computation 140 (2)*, pp. 183–202, 1998.

[2] H. Bouziri and M. Jouini, A Tabu Search Approach for the Sum Coloring Problem, *Electronic Notes in Discrete Mathematics 36 (1)*, pp. 915–922, 2010.

[3] E.K. Burke, B. McCollum, A. Meisels, S. Petrovic and R. Qu, A Graph-based Hyper-heuristic for Educational Timetabling Problems, *European Journal of Operational Research 176*, pp. 177–192, 2007.

[4] S.M. Douiri and S. Elbernoussi, New Algorithm for the Sum Coloring Problem, *International Journal of Contemporary Mathematical Sciences 6 (10)*, pp. 453–463, 2011.

[5] M.R. Garey, D.S. Johnson and H.C. So, An Application of Graph Coloring to Printed Circuit Testing, *IEEE Transactions on Circuits and Systems 23*, pp. 591–599, 1976.

[6] M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman and Company, San Francisco, 1979.

[7] C. Glass, Bag Rationalisation for a Food Manufacturer, *Journal of the Operational Research Society 53*, pp. 544–551, 2002.

[8] E. Kubicka and A.J. Schwenk, An Introduction to Chromatic Sums, *ACM Computer Science Conference*, Louisville (Kentucky), pp. 39–45, 1989.

[9] Z. Kokosinski and K. Kawarciany, On Sum Coloring of Graphs with Parallel Genetic Algorithms, *Lecture Notes in Computer Science 4431*, pp. 211–219, 2007.

[10] L.G. Kroon, A. Sen, H. Deng and A. Roy. The Optimum Cost Chromatic Partition Problem for Trees and Interval Graphs, *Lecture Notes in Computer Science 1197*, pp. 279–292, 1996.

[11] F.T. Leighton, A Graph Coloring Algorithm for Large Scheduling Problems, *Journal of Research of the National Bureau of Standards 84 (6)*, pp. 489–506, 1979.

[12] A. Lim, Q. Lou, B. Rodrigues and Y. Zhu, Heuristic Methods for Graph Coloring Problems, *Proceedings of the ACM Symposium on Applied Computing 2005*, Santa Fe, New Mexico, pp. 933–939, 2005.

[13] Y. Li, C. Lucet, A. Moukrim and K. Sghiouer, Greedy Algorithms for the Minimum Sum Coloring Problem, Logistique et transports, Mar 2009, Sousse, Tunisia. pp.LT-027.

[14] E. Malaguti and P. Toth, A Survey on Vertex Coloring Problems, *International Transactions in Operational Research 17 (1)*, pp. 1–34, 2010.

[15] N.R. Sabar, M. Ayob, R. Qu and G. Kendall, A Graph Coloring Constructive Hyper-heuristic for Examination Timetabling Problems, *Applied Intelligence 37 (1)*, pp. 1–11, 2011.

[16] D.H. Smith, S. Hurley and S.U. Thiel, Improving Heuristics for the Frequency Assignment Problem, *European Journal of Operational Research 107 (1)*, pp. 76–86, 1998.

[17] D. de Werra, C. Eisenbeis, S. Lelait and B. Marmol, On a Graph-Theoretical Model for Cyclic Register Allocation, *Discrete Applied Mathematics 93 (23)*, pp. 191–203, 1999.

[18] T.K. Woo, S.Y.W. Su and R. Newman-Wolfe, Resource Allocation in a Dynamically Partitionable Bus Network Using a Graph Coloring Algorithm, *IEEE Transactions on Communication 39*, pp. 1794–1801, 2002.

[19] Q. Wu and J.K. Hao, An Effective Heuristic Algorithm for Sum Coloring of Graphs, *Computers and Operations Research 39 (7)*, pp. 1593–1600, 2012.